Mémoire atomique auto-reconfigurable pour systèmes P2P

Vincent Gramoli

Mémoire atomique auto-reconfigurable pour systèmes P2P

Vincent Gramoli
IRISA-INRIA, Campus de Beaulieu 35042, Rennes, France.
Vincent.Gramoli@irisa.fr

Résumé : De nouvelles perspectives en terme de partage de ressources émergent des applications large échelle d'internet. Les facteurs taille et évolutivité, en termes d'insertions, départs, défaillances de noeuds, sont une difficulté supplémentaire qu'il est fondamental de prendre en compte pour traiter ce type d'applications. Une donnée partagée est une donnée qui peut être accédée de façon concurrente par plusieurs processus. Pour des raisons de tolérance aux fautes, et de performance, les données doivent être répliquées. Se pose alors le problème de maintenir une forme de cohérence entre les différentes copies d'une même donnée et ceci malgré la dynamicité des nœuds. Lorsque l'utilisateur accède à une donnée partagée, il est souhaitable que la valeur retournée lors d'une opération de lecture soit la dernière valeur écrite. Le choix du critère de cohérence permet alors de définir formellement la signification associée à l'adjectif « dernière ». Dans le cadre de notre étude, nous nous focalisons sur la linéarisabilité des opérations de lecture et écriture. Nous proposons une architecture basée sur un système de quorums qui permet (i) des lectures multiples, des écritures multiples, (ii) une répartition homogène de la charge et (iii) une auto-reconfiguration dynamique du système de quorums en fonction de la fréquence des interactions des utilisateurs. Pour cela, nous supposons que chaque nœud n'a qu'une connaissance locale du système, réduite à un voisinage proche dynamique.

Mots-clés : Systèmes distribués, Linéarizabilité, Dynamisme, Pair-à-pair, Mise à l'échelle, Durabilité, Auto-reconfiguration.

1 INTRODUCTION

Les opérations atomiques de lecture et d'écriture sont utilisées comme brique de base pour tout type d'application. De nombreux services sur internet permettent les lectures/écritures effectuées par de multiples clients. Les sites d'enchères en ligne comme *eBay* permettent aux clients de consulter l'enchère d'un produit en cours et de mettre à jour celle-ci. Également, les sites de réservation en ligne comme *voyages-sncf.com* assurent la consultation et la réservation de billets. Plus généralement, le commerce électronique à l'instar de *amazon.com* permet à des clients de consulter des prix, d'une part, et d'acheter des produits d'autre part. Ces différents services sont actuellement assurés par des serveurs centralisés relativement couteux dont la capacité est peu adaptable en vue de la sollicitation changeante de leurs ressources. Les conséquences de notre approche concerne directement la répartition et la reconfiguration dynamique de ces nombreux services face à une augmentation de charge.

1.1 Problématique

Afin d'assurer des services de qualité à des utilisateurs, il est indispensable de pallier aux problèmes liés aux défaillances impromptues. En effet, à la suite d'une panne dite « crash » du serveur, le service n'est plus assuré. Ainsi la réplication des *objets* sur plusieurs sites distants, aussi appelés *nœuds*, reste le meilleur moyen d'assurer la disponibilité de l'objet en dépit des pannes.

La réplication des données entraîne de nouveaux problèmes associés à cette distribution. Ainsi, les opérations de base – i.e. lecture et écriture – effectuées sur des réplicas distribués d'un objet doivent apparaître similaires du point de vue de tout utilisateur. De plus, une opération de lecture doit renvoyer la dernière valeur écrite de l'objet. En d'autres termes la mémoire doit être *atomique* [13]. Le modèle de cohérence mémoire introduit par Herlihy et Wing [10] appelée *linéarisabilité*, définit l'atomicité d'un objet et induit que toute exécution – i.e. séquence d'opérations effectuées sur un objet x – du système soit considérée comme une exécution séquentielle, respectant la précédence de temps réel, du point de vue des clients.

À l'heure de l'internet, un service doit être assuré indépendamment d'un nombre important de clients. Ainsi le passage à l'échelle induit une charge que doit tolérer le service. Également, les nœuds peuvent entrer dans le système, le quitter inopinément où tomber en panne fréquemment, et un mécanisme d'*auto-réparation* doit permettre de remplacer ces réplicas dynamiques au fil du temps pour assurer que suffisamment d'entre eux soient présents.

1.2 Solution

La solution proposée ici utilise, pour un objet donné, un ensemble de nœuds hébergeant une réplique de celui-ci. Ces réplicas sont regroupés au sein d'ensembles intersectés appelés *quorums de consultation* et *quorums de propagation*. Chaque opération consiste à consulter ces quorums, ainsi leur intersection impose un ordre partiel des opérations qui assure l'atomicité de l'objet, au sens de linéarisabilité. De plus, un mécanisme de réparation de ce système de quorum assure que les réplicas détectés comme défaillants soient relayés par d'autres répli-

cas. Lorsque la charge augmente au sein de ce système, celui-ci s'étend de façon autonome. Cette augmentation de charge s'explique par une diminution du nombre de réplicas utilisés et/ou par une augmentation des requêtes au système. Enfin, si la charge du système diminue alors la taille des quorums définissant la complexité des opérations diminue également.

1.3 Travaux antérieurs

La première utilisation des ensembles intersectés à des fins de cohérence mémoire est apparue avec Gifford [6] sous forme de votes valués. De nombreux résultats ont été obtenus depuis grâce à l'utilisation de ce genre d'ensembles appelés *quorums*. Citons la solution donnée par Maekawa au problème de l'exclusion mutuelle dans [14], l'émulation de mémoire partagée d'Attiya et *al*. [3] et la mémoire linéarisable acceptant des lecteurs/écrivains multiples de Lynch et Shvartsman [13].

Les problèmes liés à la dynamicité du système ont suscité un intérêt de la communauté pour la reconfiguration de quorums. Ceci aboutissant à des systèmes de quorums dynamiques. Des conditions sur la reconfiguration de quorums furent proposées par Herlihy [9]. Dans [12, 5], chaque système de quorums est vu comme une configuration et la reconfiguration y intervient comme un remplacement successif de celles-ci. Enfin, de nombreux travaux [13, 4, 7] reprennent ces changements de configurations pour pallier aux problèmes induit par le dynamisme. Dans certains cas [7], les configurations sont échangées au sein d'un ensemble fini de modèles de configurations. Dans d'autres cas [13, 7], un algorithme de consensus comme Paxos [11] assure un accord sur une configuration à installer indépendante des précédentes.

L'approche décrite précédemment consiste à redéfinir le système de quorum afin qu'un nouveau système soit utilisé, sans pour autant que la stratégie de parcours du quorum ne change. Plus récemment, des techniques permettant de modifier cette stratégie de parcours en fonction des départs et arrivés des nœuds ont été étudiées dans [1, 16, 19, 15]. Par exemple, [1] décrit un graphe De Bruijn changeant automatiquement à la suite de départs ou d'arrivée de nœuds ; [16] propose une adaptation des voisins lorsqu'un nœud quitte le système et une adaptation d'un nœud lorsqu'il est contacté par un nœud rejoignant le système. Enfin [19] propose une reconfiguration basée sur CAN [18] pour un système de quorum multi-dimensionel.

Ces deux approches sont radicalement différentes. Le caractère global des changements effectués dans la première approche nécessite que la reconfiguration soit signalée à une part non négligeable de nœuds utilisant les quorums. Le second mécanisme s'explique par l'intérêt récent attribué aux réseaux pair-à-pair (p2p) et les graphes de communication dynamiques qui y sont utilisés. Il a pour objectif de minimiser la connaissance requise de chaque nœud mais implique une détection de panne. Naor et Wieder [16] ont nommé les algorithmes appartenant à la première approche, *algorithmes non-adaptatifs*, où l'ensemble des nœuds d'un quorum à contacter est connu sans connaissance sur les nœuds potentiellement fautifs. Typiquement avant qu'un nœud contacte un quorum, il connait l'identité de l'ensemble de ses nœuds, et peut essayer de les contacter au même moment. De plus, les auteurs ont opposé à cette notion celle d'*algorithmes adaptatifs* qui s'adaptent en fonction des départs et arrivés des nœuds. Cette notion implique que chaque nœud d'un quorum soit contacté tour-à-tour. Ici, nous nous intéressons essentiellement aux problèmes de passage à l'échelle dans des réseaux de pairs, c'est pourquoi la connaissance de chaque nœud est bornée et nous utilisons une approche adaptative.

1.4 Contributions

Nous proposons un algorithme assurant l'atomicité des opérations dans un système distribué hautement dynamique où le mode de communication est asynchrone. De plus, notre solution est tolérante aux défaillances et vise à assurer un bon compromis entre charge et complexité.

1.4.1 Tolérance aux défaillances

La valeur de l'objet est répliquée sur un ensemble de nœuds du système. Ainsi une certaine flexibilité est accordée au nombre de pannes. Il peut arriver que l'ensemble des réplicas tombe en panne. C'est pourquoi nous utilisons un système de réplication active, qui introduit de nouveaux réplicas en réponse aux pannes. Ce mécanisme dépend directement des pannes, de la taille et de la charge du système.

1.4.2 Équilibrage de charge

Chacun des nœuds maintient une connaissance restreinte du système. Il conserve seulement l'information (par exemple l'identité) d'un nombre borné de ses voisins dans la topologie pour pouvoir communiquer. Comme expliqué dans [16], la charge en est d'autant diminuée puisque l'espace mémoire alloué est moins important. Comme nous le verrons dans la section 3, chaque nœud effectue une opération de lecture ou d'écriture sur un objet en consultant des ensembles de réplicas, les quorums. Nous supposons que ces requêtes sont effectuées de façon aléatoire en tout point de notre système de quorums. Lorsqu'un grand nombre de nœuds effectuent des opérations sur un même objet dans une même période de temps, les réplicas de cet objet sont fortement sollicités. Nous proposons un mécanisme d'extension du système de quorums utilisé pour diminuer la charge de chaque réplica en augmentant leur nombre et assurant l'équilibrage de la charge au sein du système. Plus précisément, on ajoute des réplicas aux quorums en minimisant la taille des intersections. Ainsi le système de quorum est augmenté et la charge est mieux répartie.

1.4.3 Complexité de consultation

Il est important de noter que le taux de requêtes faîtes par les nœuds du réseau de pairs sont variables. Par exemple,

en fonction de l'heure de la journée, une plus ou moins importante part de nœuds sont actifs et donc capables d'effectuer des opérations. Ainsi, après avoir diminuer la charge en augmentant le système de quorums, il se peut que le taux de requêtes diminuent à nouveau. Dans ce cas, le nombre élevé de quorums et la propriété d'intersection minimale de ceux-ci implique une taille de ces quorums élevée. On obtient donc une complexité élevée en nombre de messages pour contacter tous les nœuds d'un quorum, lors d'une opération. La charge étant faible de nouveau, il est important de diminuer la complexité en réduisant le système de quorums. Nous proposons un mécanisme de réduction de la complexité lorsque la charge diminue de nouveau.

1.5 Plan

Dans la seconde partie de cet article nous présentons le modèle. Dans la section 3, nous présentons le fonctionnement des opérations tandis que la section 4 introduit des solutions aux problèmes liés aux défaillances. Les mécanismes utilisés pour répartir la charge sont présentés dans la section 5. Enfin, la section 6 présente quelques perspectives de recherche et les travaux en cours.

2 MODÈLE

Le système est un ensemble de processus appelés *nœuds*. Il est représenté par un graphe de communication faiblement connecté, où les nœuds peuvent communiquer par l'intermédiaire de liens. On définit une couche abstraite consistant en un sous-ensemble de ces nœuds qui connaissent l'identité de leurs voisins. Chacun de ces nœuds hébergent une copie de l'objet considéré et une *estampille* (renseignant sur la version de la valeur en cours de l'objet) et sont appelés *réplicas*. Cette abstraction est représentée par un tore, où les réplicas sont des sommets et un arc existe entre eux si ils sont voisins.

2.1 Modèle de fautes

Le dynamisme du système s'exprime par des arrivés, des départs et des pannes. Ces pannes sont « crash », tout nœud peut joindre le système, quitter le système ou tomber en panne – tout nœud sortant d'un état de panne étant considéré comme un nouveau nœud entrant. Tout nœud du système qui n'est pas tombé en panne et qui n'a pas quitté le système est considéré comme *actif*, sinon il est dit *inactif*. Enfin, le modèle de communication est asynchrone les messages peuvent être perdus, réordonnés et arbitrairement retardés.

2.2 Topologie

Pour chaque objet un espace de coordonnées réelles est partagé au sein des réplicas présents dans le système à la manière de CAN [18]. Nous nous intéressons ici à un seul objet et donc à un seul système de quorums. Une plus large mémoire s'étend trivialement comme

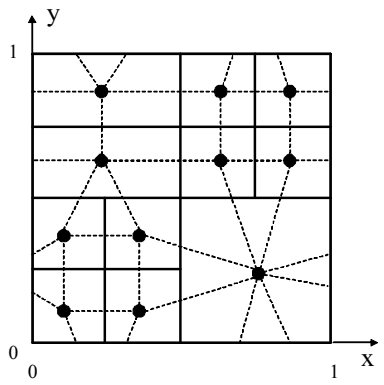

FIG. 1 – La topologie en tore : L'espace de coordonnées est découpé en zones. Chaque zone possède un réplica (représenté par un point) responsable. Deux réplicas sont voisins (liés par des pointillés) si les zones dont ils sont responsables sont adjacentes.

composition de ces objets atomiques. Dans notre cas, on se donne un espace bidimensionnel $[0,1) \times [0,1)$ dont le premier nœud détenteur de l'objet est initialement responsable. Au fur et à mesure de l'arrivée de nouveaux nœuds, l'espace est de plus en plus divisé. À contrario, les départs (ou pannes) de nœuds entraînent une fusion de ces sous-espaces appelés *zones*, et l'union de ces zones représente à tout moment l'espace $[0,1) \times [0,1)$. Deux nœuds sont dits *voisins* si ils sont responsables de zones adjacentes. La figure 1 représente un tore $[0,1) \times [0,1)$ divisés en de multiples zones.

Chaque nœud est représenté par un unique identifiant. On note l'ensemble de ces identifiants $I \subset N$. L'ensemble des valeurs que peut prendre l'objet est noté V et l'ensemble des estampilles associées est T. Chaque réplica conserve une certaine vision de l'objet, c'est-à-dire qu'il conserve la dernière valeur (selon lui) de l'objet et l'estampille associée à l'écriture correspondante. Une estampille est associée à chaque valeur écrite, et est monotoniquement incrémentée à chaque nouvelle écriture. Ainsi l'estampille définit un ordre total sur les opérations d'écritures effectuées. Afin de différencier deux écritures concurrentes effectuées depuis différents nœuds, l'estampille possède un entier de poids faible correspondant à l'identifiant du nœud l'exécutant. Ainsi l'estampille est un couple $\langle compteur, id\_noeud \rangle \in N \times I$.

Le tore utilisé a pour borne b telle que $b.xmin = b.ymin = 0$ et $b.xmax = b.ymax = 1$. Chaque réplica de la mémoire atomique est responsable d'une zone bornée. Cette zone z est délimitée par un rectangle défini par un point inférieur gauche de coordonnées $(z.xmin, z.ymin)$ et un point supérieur droit de coordonnées $(z.xmax, z.ymax)$. La zone b attribuée au premier nœud correspond donc à l'ensemble des points dont l'abscisse est dans l'intervalle $[b.xmin, b.xmax)$ et l'ordonnée est dans l'intervalle $[b.ymin, b.ymax)$. On utilisera la notation $[z.xmin, z.xmax) \times [z.ymin, z.ymax)$

pour caractériser la zone z.

3 OPÉRATIONS ATOMIQUES

Les opérations considérées sont des opérations de lecture et d'écriture. Elles sont divisées en deux phases, à l'instar des opérations non-adaptatives de [3, 13]. Ces phases sont appelées phase de consultation et phase de propagation. Une optimisation présente dans [2] permet d'obtenir une opération de lecture effectuée en une seule phase. Par manque de place, nous ne présentons pas cette amélioration dans cet article.
Une écriture s'effectue seulement sur certains nœuds du système de quorums. Cela implique qu'il existe des nœuds qui ne sont pas à jour lorsqu'une écriture termine. Cependant le protocole d'une opération de lecture assure qu'au moins un nœud à jour soit consulté. Afin de détecter ce nœud parmi l'ensemble des nœuds consultés, il est important d'associer une estampille à la valeur, indiquant son numéro de version : soient e_1 et e_2 deux estampilles associées respectivement aux valeurs v_1 et v_2, si $e_1 < e_2$ alors la valeur v_2 associée est plus à jour que la valeur v_1.

Chaque quorum de consultation intersecte tout quorum de propagation. Ainsi une consultation informe l'initiateur au sujet de la dernière valeur propagée sur n'importe quel quorum de propagation. Cette phase est utile à la lecture pour connaître la valeur à renvoyer, mais elle permet aussi à l'écriture de récupérer la plus grande estampille du système. Ensuite, la phase de propagation permet d'assurer que lorsqu'une opération termine, toute opération suivante ne prendra pas en compte une ancienne valeur ou une estampille non maximale. Si l'opération est une lecture, cette dernière phase propage la paire $\langle$ estampille, valeur $\rangle$ lue à la phase précédente. Dans le cas d'une écriture, cette phase propage la valeur à écrire avec une nouvelle estampille plus grande que celle consultée à la phase précédente. Ces phases sont détaillées dans les sous-sections 3.3 et 3.4.

3.1 Routage

Chaque nœud a connaissance de ses voisins. Ainsi chaque nœud conserve l'identité de ces derniers et a la capacité de les contacter. Le routage permet de définir un chemin suivant lequel une information transite. Dans le cas d'une phase de consultation, le routage assure que la plus grande estampille et la valeur à jour associée rencontrées, sont transmises de voisin en voisin jusqu'à l'initiateur suivant un anneau horizontal du tore. La phase de propagation quant à elle, permet de propager la dernière valeur et son estampille maximale associée aux éléments constituant un anneau vertical du tore. Ces valeur, estampille proviennent de la précédente phase.

Néanmoins, il peut arriver qu'un message soit perdu, ou qu'un des nœuds contactés tombe en panne. Pour pallier à une perte éventuelle de message, l'initiateur envoie régulièrement les messages d'une phase tant qu'il n'a pas appris sa terminaison. Dans le cas où un nœud à contacter tombe en panne, cela revient à des pertes de message avant qu'un voisin prenne la responsabilité de sa zone. Comme détaillé dans les sections suivantes, un des voisins prend la responsabilité de la zone du réplica fautif et en informe ses voisins. Dans le cas où l'initiateur tombe en panne, il est possible que l'information sur la requête reçue soit perdue. Ainsi une opération peut ne pas terminer. À noter que l'atomicité fait l'objet d'une propriété de sûreté et il est nécessaire de supposer que les opérations initiées terminent pour qu'elles soient linéarisables. Enfin l'étude temporelle de ce protocole ne fait pas l'objet de cet article et nous ne présentons pas de résultats sur la progression où la vivacité de celui-ci.

3.2 Quorums dynamiques

Dans le contexte des quorums en grille, un quorum est défini comme une ligne ou une colonne. Ainsi il existe deux types de quorums, et n'importe quel quorum d'une ligne intersecte n'importe quel quorum d'une colonne. Dans notre contexte, chaque zone découpant l'espace bidimensionel n'est pas nécessairement carrée. C'est pourquoi nous définissons deux types de quorums légèrement différents. Un quorum de consultation (resp. de propagation) est un ensemble de nœuds définis par une traversée horizontale (resp. verticale) effectuée par le routage précédemment cité.

Un quorum regroupe un ensemble d'éléments en fonction de leur zone. Ainsi les éléments constituant les quorums dépendent des limites de leur zone au moment où ils sont contactés. Le dynamisme de ces zones induit un dynamisme dans le choix des nœuds d'un quorum. Autrement dit, si un même initiateur est contacté pour la même phase à deux instants différents, les quorums qu'il contacte ne sont pas nécessairement identiques du fait des départs et arrivées de nouveaux nœuds. Les quorums utilisés ici sont dit *dynamiques*.

Pour des raisons évidentes de simplicité les définitions de quorums suivantes ne mentionnent pas de paramètres temporels induit par le dynamisme.
Un quorum de consultation, comme son nom l'indique, est un ensemble de nœuds qui sont consultés afin d'obtenir la valeur et l'estampille qu'ils conservent de l'objet.

Definition 1 (Quorum de consultation Q_c) *Un quorum de consultation $Q_c \subset I$, est l'ensemble des nœuds responsables des zones de Z_c telles que*
- $\bigcup_{\forall z \in Z_c} \{[z.xmin, z.xmax)\} = [b.xmin, b.xmax)$
- $\bigcap_{\forall z \in Z_c} \{[z.xmin, z.xmax)\} = \emptyset$
- $\exists z \in Z_c, \forall z' \in Z_c, z.ymin + (z.ymax - z.ymin/2) \in [z'.ymin, z'.ymax)$

Un quorum de propagation est un ensembe de nœuds sélectionnés pour mémoriser l'écriture d'une nouvelle valeur. Ces nœuds mettent à jour la valeur de l'objet et l'estampille associée qu'ils conservent.

Definition 2 (Quorum de propagation Q_p) *Un quorum de propagation $Q_p \subset I$, est l'ensemble des nœuds responsables des zones de Z_p telles que*

– $\bigcup_{\forall z \in Z_p}\{[z.ymin, z.ymax)\} = [b.ymin, b.ymax)$
– $\bigcap_{\forall z \in Z_p}\{[z.ymin, z.ymax)\} = \emptyset$
– $\exists z \in Z_p, \forall z' \in Z_p, z.xmin + (z.xmax - z.xmin/2) \in [z'.xmin, z'.xmax)$

La classification des quorums en deux types, où tout quorum d'un type intersecte ceux de l'autre type, a été initialement proposée par Herlihy [8].

3.3 Phase de consultation

Tout nœud du système peut initier une opération de lecture ou d'écriture à n'importe quel moment sous réserve que toute opération initiée avant ait terminé. Ce nœud doit avoir joint le système et y être toujours actif mais peut également être un réplica d'un des quorums de la mémoire atomique.

Les deux types d'opérations commencent par une phase de consultation où un quorum de consultation est contacté. Afin que chaque nœud puisse initier une opération, on émet l'hypothèse qu'une fonction prenant les coordonnées réelles d'un point permet de retrouver le responsable de la zone le contenant. Ainsi le nœud peut prendre connaissance d'un réplica arbitrairement choisi et lui envoyer la requête de l'opération.

Tout nœud de ce système de quorum appartient à au moins un quorum de consultation d'après la définition 1. Ainsi lorsqu'une requête d'opération parvient à un tel réplica i, appelé l'*initiateur*, i propage selon une traversée horizontale un message contenant (i) son estampille, (ii) sa valeur et (iii) sa zone. Étant donnée que chaque réplica connait ses voisins directs, le message est propagé de voisin en voisin dans un sens horizontal comme indiqué sur la figure 2. Il est possible que plusieurs réplicas soient voisins dans cette direction. Dans ce cas, la traversée est déterminée par une fonction prenant en argument les limites verticales de la zone de l'initiateur ($i.ymin$ et $i.ymax$). Puis j est choisi parmi les voisins dans le sens considéré tel que l'ordonnée du centre de la zone de i appartienne à l'intervalle verticale de la zone de j. Formellement on obtient j tel que :

$$i.ymin + (i.ymax - i.ymax)/2 \in [j.ymin, j.ymax)$$

À chaque fois qu'un nœud reçoit ce message il inspecte le contenu du message. S'il contient une estampille plus grande ou égale à la sienne alors il propage le même message, sinon cela signifie qu'il a connaissance d'une valeur plus à jour ; dans ce cas il remplace l'estampille et la valeur du message par les siennes. Du fait de cette fonction déterministe et de la topologie en anneau parcourue, il est inévitable que l'initiateur i reçoive le message qu'il avait le premier envoyé. De plus, comme l'estampille est mise à jour (si nécessaire) à chaque réception du message, l'initiateur récupère la plus grande estampille et la dernière valeur dont tout élément du quorum de consultation a connaissance. Dans le cas d'une opération d'écriture, l'entier de l'estampille rencontrée est incrémentée, son identifiant devient l'identifiant de celui qui a émis la requête et la valeur devient celle à écrire. Dans le cas d'une opération de lecture la paire ⟨ estampille, valeur ⟩ reste inchangée. Dans les deux cas, la paire résultante est celle utilisée dans la phase de propagation et son estampille définit l'ordre de l'opération.

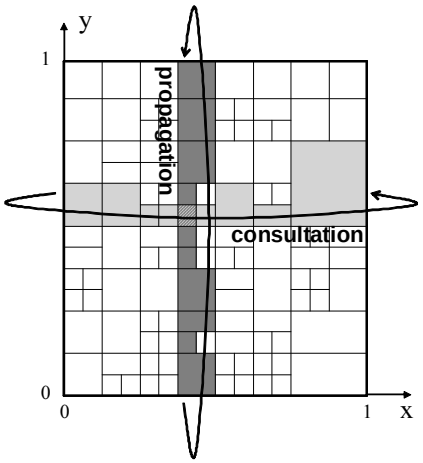

FIG. 2 – Les phases de traversées : Ici sont représentées deux traversées. La première, horizontale, indique les nœuds d'un quorum de consultation contactés et est appelée phase de consultation. La seconde, verticale, indique un quorum de propagation et est appelée phase de propagation.

3.4 Phase de propagation

La seconde phase des opérations est effectuée de façon analogue. L'initiateur envoie un message correspondant à cette phase et contenant la paire ⟨ estampille, valeur ⟩, dans un sens vertical bien précis. La traversée est effectuée par une retransmission de voisin en voisin du même message. À la différence de précédemment, le message n'est pas modifié mais la paire ⟨ estampille, valeur ⟩ conservée localement par chacun des participants est modifiée si une plus récente est reçue (i.e., une estampille plus grande est reçue). Une traversée est effectuée selon un axe vertical de telle manière que tout élément d'un quorum de propagation ait la paire ⟨ estampille, valeur ⟩ à jour.

Claim 3.1 *L'ordre partiel des opérations défini par l'estampille assure la propriété d'atomicité.*

La preuve d'atomicité d'un seul objet dépend essentiellement de l'intersection non vide des deux types de quorums. L'atomicité de la mémoire résulte de la composition des objets atomiques.

4 LA TOLÉRANCE AUX DÉFAILLANCES

Pour que notre service soit efficace en terme de *qualité de service*, il faut que la disponibilité des données soit toujours assurée. Ainsi le service doit être tolérant aux défaillances.

4.1 Réplication des données

Lorsque les données sont centralisées, une panne du système entraîne une indisponibilité du service associé. Certains serveurs adoptent une approche répliquée en utilisant des serveurs-mirroirs. Cette approche ne résoud pourtant pas les problèmes liés aux variations incontrôlables de la charge du système. La particularité d'un grand système est le nombre important de participants potentiels qu'il contient. Notre réplication s'effectue au sein d'un ensemble de ces nœuds participants. Ce regroupement au sein d'un réseau de pairs peut être basé sur un recoupement de centres d'intérêt : sachant que deux nœuds possèdent des objets similaires il est probable, que la réplication d'un objet du premier nœud sur le second profite à ce dernier. Nous laissons cette remarque de côté, celle-ci ne faisant pas partie de l'objet de cet article.

Ainsi un certain ensemble de nœuds possèdent une réplication de la donnée et si une partie stricte d'entre eux tombent en panne au même instant, l'objet reste accessible en dépit des pannes. Comme mentionné dans la sous-section 3.2, chaque opération consiste à contacter des quorums. Ainsi le service est assuré si et seulement si le quorum contacté est disponible. Or, la définition de nos quorums et l'auto-adaptation de notre système assure l'existence d'un quorum de consultation et un quorum de propagation s'il reste au moins un réplica actif.

Également, lorsque les pannes s'accumulent un mécanisme d'auto-réparation assure le remplacement de réplicas par une réplication active au sein de nœuds extérieurs.

4.2 Détection de défaillances

La détection de défaillance permet au système de s'auto-adapter pour que les quorums nécessaires aux opérations restent disponibles. En effet lorsqu'un nœud tombe en panne, sa zone n'a plus de responsable. Ainsi il existe des nœuds du système de quorums d'où part une traversée qui n'aboutira pas. Ces traversées sont celles qui doivent passées par le nœud fautif. L'inactivité de ce nœud bloque ainsi la traversée.

Pour pallier à ce problème un mécanisme de détection de faute est utilisé. Celui-ci repose sur un échange de messages périodiques, qualifiés de *battements de cœur*. L'absence de réception d'un tel message durant une certaine période de temps indique au récepteur que l'émetteur est tombé en panne. Chacun des messages de battement de cœur sert également à mettre à jour la table des voisins, c'est-à-dire pour chacun d'entre eux, leur zone, les identifiants de leurs voisins, l'estampille et la valeur qu'ils conservent.

4.3 Arrivée d'un nœud

On suppose qu'à la création du système un seul nœud est détenteur de l'objet. Ainsi il crée son système de quorum tel qu'il soit l'unique responsable de toute la zone $[b.xmin, b.xmax) \times [b.ymin, b.ymax)$.

Lorsqu'un nœud décide de joindre le système, il contacte un nœud déjà présent, attend son accusé-réception et devient actif. À ce stade le nœud est seulement un candidat potentiel pour faire partie du système de quorum. Ce n'est que lorsque le système décide de s'étendre par le mécanisme d'auto-reconfiguration, qu'un réplica i choisi parmi les candidats potentiels, un nœud j pour y forcer la réplication. Puis i partage sa zone en deux parties égales, et en attribue une au nouveau réplica j. Il informe j de la zone dont il est responsable et de l'identité des réplicas dont les zones sont adjacentes à celle de j, i.e., ses voisins. Cette intégration d'un nouveau nœud est détaillée dans la sous-section 5.3.

4.4 Départ d'un nœud

Il est important de constater qu'une traversée ou une diagonalisation (cf. sous-section 5.2) peut être retarder entre le moment où un voisin concerné tombe en panne et le moment où le voisinnage est redéfinit. Le protocole qui suit est similaire à celui proposé dans la table de hachage CAN [18]. Lorsqu'un réplica i est considéré comme étant en panne, le système décide de s'auto-reconfigurer afin de permettre aux quorums d'être toujours accessibles. Pour cela un nœud voisin ne recevant pas de messages de *battements de cœur* de la part de i le considère comme fautif. À noter qu'il existe plusieurs voisins détectant cette panne. Un voisin j est choisi parmi ceux-ci selon un critère simple : le nœud ayant découpé sa zone en dernier est le nœud choisi. À noter qu'un nœud peut éventuellement être responsable de plusieurs zones afin d'assurer que leur forme reste rectangulaire (cf. [17] pour plus de détails). Ensuite la zone dont le réplica choisi est responsable, est étendue avec la zone du réplica qui a quitté le système. Le réplica considéré connait l'identité des voisins du nœud parti, et contacte ses voisins diagonaux pour récupérer la paire $\langle$ estampille, valeur $\rangle$ (ou une plus à jour) des voisins verticaux. Enfin si l'estampille découverte est plus récente, il change sa paire locale en celle découverte et informe ses voisins de sa nouvelle zone. Le mécanisme de traversée ou de diagonalisation éventuellement en attente est alors poursuivi.

5 RÉPARTITION DE LA CHARGE

Chacun des réplicas présents dans le système de quorums subit une charge due aux nombres de requêtes qu'il reçoit durant une courte période de temps. D'après le fonctionnement des opérations, on remarque que lorsqu'un réplica effectue une opération en tant qu'initiateur, il provoque une charge quasiment équivalente sur chacun des membres de son quorum. Ainsi si la charge d'un nœud est c à un instant, on sait que $\Omega(c)$ sera attribué à chacun des nœuds des quorums auxquels il appartient dans le cas où toute requête est traitée.

5.1 Charge et Complexité

Il existe un compromis entre la charge et la complexité du système. Lorsque les quorums contiennent en moyenne un grand nombre de réplicas, la complexité en nombre de

messages d'une traversée est plus importante. Cependant lorsque leur taille moyenne est petite, cela signifie que la charge globale du système est répartie sur un nombre moins important de réplicas et la charge moyenne d'un réplica est d'autant plus grande. Ici, nous nous efforçons d'utiliser des quorums dont l'intersection avec un autre quorum contient exactement un réplica. Ainsi, à condition d'avoir une stratégie d'accès uniforme en tout point du système de coordonnées, le système de quorums atteint n réplicas avec des quorums de taille $O(\sqrt{n})$. Et, la complexité de contact d'un quorum – définie par le temps et le nombre de messages nécessaires pour contacter un quorum – est optimale comme montré dans [16].

Un problème se pose lorsque la charge d'un nœud devient trop importante. Cela signifie que le taux de requêtes reçues devient supérieur au taux de traitement des opérations du nœud. On ajoute au modèle précédent une file permettant à un réplica de stocker les requêtes reçues en attendant de les traiter. Après un certain délai de garde ces requêtes sont traitées. Un nombre seuil de requêtes enregistrées définit une surcharge du réplica. Ce nombre reste inférieur à la capacité de la queue.

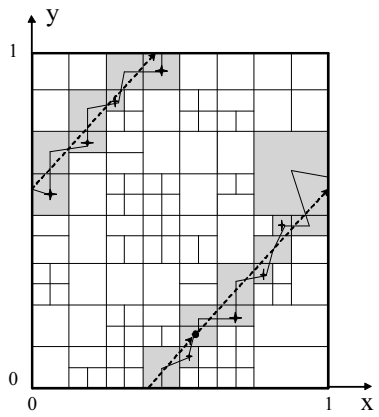

FIG. 3 – La diagonalisation : Les zones grisées sont celles des réplicas impliqués dans la diagonalisation. Le point indique l'initiateur de la diagonalisation et les croix représentent les autres réplicas inspectant leur charge. Dans ce scénario tous les réplicas répréstentés par des croix ou un point sont surchargés. Ainsi la diagonalisation s'achève seulement lorsque le message retourne a l'initiateur.

5.2 Diagonalisation

Si un réplica surchargé reçoit une requête alors il transmet directement la requête à un autre nœud. D'après la remarque précédente, on sait qu'il est inutile pour un réplica i d'envoyer la requête à un membre j d'un même quorum. En effet, si j accepte de traiter la requête, il a davantage de chance d'être surchargé puisque i le sollicitera pour les opérations dont certaines requêtes sont dans sa queue. Réciproquement, j acceptant cette requête pourra éventuellement demander la participation de i dans l'opération, entraînant une augmentation de la charge de i celui-ci étant déjà surchargé.

Afin de contacter des quorums différents nous définissons un parcours diagonal parmi les réplicas du système de quorums. De plus, afin d'assurer la détection d'un système surchargé, ce parcours suit une diagonale parallèle à celle traversant le tore du point $(xmin, ymin)$ au point $(xmax, ymax)$. Ainsi tout réplica dont la zone est traversée par cette droite, est contacté. Si le réplica contacté est un voisin indirect du dernier nœud à avoir tester sa charge, alors il vérifie sa charge à son tour. S'il n'est pas surchargé, il insère la requête dans sa queue. Sinon il retransmet la requête et le scénario est répété. Cette stratégie est représentée sur la figure 3.

5.3 Auto-expansion

Comme le mécanisme poursuit sur une trajectoire parallèle à la diagonale, l'initiateur de cette diagonalisation reçoit finalement le message si aucun des réplicas contactés ne décide d'initier l'opération. Ainsi, l'initiateur sait que tous les replicas testés étaient surchargés. Dans ce cas, une approximation sur la charge apparemment élévée du système est faite et l'initiateur décide d'étendre le système de quorums.

Lorsqu'un réplica i décide d'étendre le système de quorum, il fait appel à un nœud du système hors du système de quorum. Autrement dit, un nœud actif mais n'étant pas membre des quorums. Il est raisonnable de supposer l'existence d'un tel nœud car la charge du système de quorum est proportionnelle au nombre total de nœuds actifs par rapport au nombre de réplicas du système de quorums. Or, si les nœuds actifs du système sont uniquement les réplicas membres des quorums et que le système est en phase d'expansion cela signifie que les réplicas suffisent par leur requête à se surcharger eux-mêmes. Plus précisément, avec $\tau_r(p)$, le taux de requêtes que peut délivré un nœud pendant une période p, $\tau_t(p)$ le taux maximal de traitement des requêtes effectuées par un nœud durant la même période et tq la taille maximale d'un quorum, alors une surcharge durable n'est possible que lorsque $\tau_r(p) > \frac{\tau_t(p)}{tq}$. Le terme « durable » est utilisé pour pallier à l'irrégularité dans la réception des messages induit par l'asynchronisme. Ce terme signifie durant une période assez longue pour que tout message non perdu arrive à destination. Dans la suite, on suppose l'existence d'un tel nœud lorsqu'une expansion a lieu.

Une fois que i trouve un nœud j actif mais non membre du système de quorums, i réplique l'objet en copiant la valeur de l'objet qu'il possède et l'estampille associée en j. Il informe également j qu'il fait partie de ses voisins. Alors i découpe sa zone en deux parties égales selon un axe horizontal. Ce choix est effectué par simplicité mais il peut s'agir d'une découpe variable afin de favoriser la rapidité de certaines opérations comme présenté dans [2]. Une des deux zones obtenues, est alors attribuée à j, et i met à jour sa zone de responsabilité avec la seconde zone obtenue. L'identité des voisins de i devenus voisins de j du fait de cette découpe est également transmise à j par i.

5.4 Auto-réduction

Si pendant une trop longue période de temps, définie préalablement en fonction de l'application, aucune requête n'a été effectuée sur un réplica, ce réplica peut décider de provoquer une réduction du système de quorums afin de diminuer la complexité de contact d'un quorum. Pour cela le réplica concerné cherche directement parmi ses voisins le dernier à avoir divisé sa zone et l'informe de son départ. L'auto-reconfiguration s'effectue comme mentionné dans 4.4 une fois que la faute a été détectée.

6 CONCLUSION

Nous avons présenté ici, l'intégration de différentes techniques utilisées dans les systèmes distribués. Notamment les couches de communication utilisées dans les réseaux p2p, la réplication contrôlée et les quorums dynamiques. Nous avons proposé des solutions aux problèmes de cohérence mémoire, de tolérance aux défaillances, de complexité de consultation et de charge du système comme composantes d'une mémoire atomique dynamique pour grands systèmes et plus précisément réseaux de pairs.

Il n'a pas été explicitement mentionné comment conserver un nombre minimal de réplicas au sein de la mémoire atomique. Il serait intéressant de trouver un état minimal en fonction du taux de pannes du système.
Actuellement un travail est effectué sur la formalisation de ces idées et la preuve d'atomicité des opérations. Certaines améliorations telles que la simplification des opérations sont à l'étude. Une implémentation et une analyse de résultats sont à venir.
Dans un cadre plus large, il serait d'une part intéressant de donner un critère de similarité entre réplicas justifiant la réplication active d'un objet sur tel ou tel nœud. D'autre part, on pourrait étendre ce travail en prenant en compte le fait qu'un réplica puisse sortir de son état de panne, et étudier dans ce cas les conditions assurant l'unicité d'un système de quorums pour un objet donné.

BIBLIOGRAPHIE

[1] Ittai Abraham and Dahlia Malkhi. Probabilistic quorums for dynamic systems. In Faith Ellen Fich, editor, *Distributed algorithms*, volume 2848/2003 of *Lecture Notes in Computer Science*, pages 60–74, Oct 2003.

[2] Emmanuelle Anceaume, Maria Gradinariu, Vincent Gramoli, and Antonino Virgillito. SAM : Self-* atomic memory for P2P systems. Technical Report 1717, IRISA, Rennes, June 2005.

[3] Hagit Attiya, Amotz Bar-Noy, and Danny Dolev. Sharing memory robustly in message-passing systems. *J. ACM*, 42(1) :124–142, 1995.

[4] Shlomi Dolev, Seth Gilbert, Nancy Lynch, Alex Shvartsman, and Jennifer Welch. GeoQuorums : Implementing atomic memory in ad hoc networks. In *Proc. of 17th International Symposium on Distributed Computing*, pages 306–320, 2003.

[5] Burkhard Englert and Alex Shvartsman. Graceful quorum reconfiguration in a robust emulation of of shared memory. In *Proc. of Int. Conf. on Distributed Computing Systems (ICDCS 2000)*, pages 454–463, 2000.

[6] David K. Gifford. Weighted voting for replicated data. In *SOSP '79 : Proceedings of the seventh ACM symposium on Operating systems principles*, pages 150–162. ACM Press, 1979.

[7] Seth Gilbert, Nancy Lynch, and Alex Shvartsman. RAMBO II : Rapidly reconfigurable atomic memory for dynamic networks. In *Proc. of 17th Int. Conference on Dependable Systems and Networks*, pages 259–269, 2003.

[8] Maurice Herlihy. A quorum-consensus replication method for abstract data types. *ACM Trans. Comput. Syst.*, 4(1) :32–53, 1986.

[9] Maurice Herlihy. Dynamic quorum adjustment for partitioned data. *ACM Trans. Database Syst.*, 12(2) :170–194, 1987.

[10] Maurice P. Herlihy and Jeannette M. Wing. Linearizability : a correctness condition for concurrent objects. *ACM Trans. Program. Lang. Syst.*, 12(3) :463–492, 1990.

[11] Leslie Lamport. The part time parliament. *ACM Transactions on Computer Science*, 16(2) :133–169, 1998.

[12] Nancy Lynch and Alex Shvartsman. Robust emulation of shared memory using dynamic quorums. In *Proc. of 27th Int. Symp. on Fault-Tolerant Comp.*, pages 272–281, 1997.

[13] Nancy Lynch and Alex Shvartsman. RAMBO : A reconfigurable atomic memory service for dynamic networks. In *Proc. of 16th International Symposium on Distributed Computing*, pages 173–190, 2002.

[14] Mamoru Maekawa. A $\sqrt{N}$ algorithm for mutual exclusion in decentralized systems. *ACM Trans. Comput. Syst.*, 3(2) :145–159, 1985.

[15] Uri Nadav and Moni Naor. Fault-tolerant storage in a dynamic environment. In *DISC '04 : The 18th Annual Conference on Distributed Computing*, 2004.

[16] Moni Naor and Udi Wieder. Scalable and dynamic quorum systems. In *PODC '03 : Proceedings of the twenty-second annual symposium on Principles of distributed computing*, pages 114–122. ACM Press, 2003.

[17] Sylvia Ratnasamy. *A Scalable Content-Addressable Network*. PhD thesis, University of California at Berkeley, October 2002.

[18] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard M. Karp, and Scott Shenker. A scalable content-addressable network. In *SIGCOMM*, pages 161–172, 2001.

[19] Bujor Silaghi, Pete Keleher, and Bobby Bhattacharjee. Multi-dimensional quorum sets for read-few write-many replica control protocols. In *In Proc. of the 4th CCGRID/GP2PC*, April 2004.